12-2018

Utilizing computational trust to identify rumor spreaders on Twitter

Bhavtosh RATH

Wei GAO
Singapore Management University, weigao@smu.edu.sg

Jing MA

Jaideep SRIVASTAVA

Published in Social Network Analysis and Mining,
December 2018, Volume 8, Issue 64, Pages 1-16
https://doi.org/10.1007/s13278-018-0540-z

Utilizing computational trust to identify rumor spreaders on Twitter

Bhavtosh Rath[1] · **Wei Gao**[2] · **Jing Ma**[3] · **Jaideep Srivastava**[1]

Abstract
Ubiquitous use of social media such as microblogging platforms opens unprecedented chances for false information to diffuse online. Facing the challenges in such a so-called "post-fact" era, it is very important for intelligent systems to not only check the veracity of information but also verify the authenticity of the users who spread the information, especially in time-critical situations such as real-world emergencies, where urgent measures have to be taken for stopping the spread of fake information. In this work, we propose a novel machine-learning-based approach for automatic identification of the users who spread rumorous information on Twitter by leveraging computational trust measures, in particular the concept of *Believability*. We define believability as a measure for assessing the extent to which the propagated information is likely being perceived as truthful or not based on the proxies of trust such as user's retweet and reply behaviors in the network. We hypothesize that the believability between two users is proportional to the trustingness of the retweeter/replier and the trustworthiness of the tweeter, which are complementary to one another for representing user trust and can be inferred from trust proxies using a variant of HITS algorithm. With the trust network edge-weighted by believability scores, we apply network representation learning algorithms to generate user embeddings, which are then used to classify users into rumor spreaders or not based on recurrent neural networks (RNN). Experimented on a large real-world rumor dataset collected from Twitter, it is demonstrated that our proposed RNN-based method can effectively identify rumor spreaders and outperform four more straightforward, non-RNN models with large margin.

Keywords Rumor detection · Computational trust · Representation learning · Recurrent neural networks

1 Introduction

In recent years, social media platforms have been increasingly witnessed as an emerging sphere for generating and spreading false or unverified information. For example, the news about Facebook CEO Mark Zuckerberg offering money to Facebook users who do not share social media hoaxes is itself a parody of social media hoaxes.[1] False rumors can be potentially detrimental, triggering serious repercussions or consequence in our society. A rumor circulating on Facebook and Twitter since December 5, 2015 claimed that Muslim residents of Dearborn, Michigan, held a pro-ISIS march, where protesters were carrying ISIS flags.[2] This rumor was circulated following the mass shooting in San Bernardino, California, by a U.S.-born Muslim who became radicalized while living in the U.S. and whose wife was from Pakistan. On a daily basis, such misinformation originates and propagates within social media outlets, rendering the quality and credibility of social media content seriously inferior.

Social psychology literature defines rumor as a story or a statement whose truth value is unverified or deliberately false (Allport and Postman 1965). Differentiating rumor from fact, or measuring the truthfulness of information

✉ Wei Gao
wei.gao@vuw.ac.nz

Bhavtosh Rath
rathx082@umn.edu

Jaideep Srivastava
srivastava@umn.edu

[1] Department of Computer Science and Engineering, University of Minnesota, Twin Cities, Minneapolis, MN, USA

[2] School of Information Management, Victoria University of Wellington, Wellington, New Zealand

[3] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong

[1] https://www.snopes.com/zuckerberg-dont-share-hoaxes/.

[2] http://www.factcheck.org/2015/12/dearborns-anti-isis-rally/.

directly is technically very challenging. While one way to address this is by debunking the false information using rumor detection and classification approach (Castillo et al. 2011; Liu et al. 2015; Ma et al. 2015, 2016, 2017, 2018a; Yang et al. 2012; Wu et al. 2015; Zhao et al. 2015), another way is by estimating whether the spreader of concerned information is trusted or not and to what extent by their peers so as to identify the "high-risk" users who are more likely to spread false information online. These high-risk users are effectively labeled so that other online users can be drawn appropriate attention or be altered against the credibility of information posted by the labeled users. To the best of our knowledge, there is no existing approach that has fallen into this second category based on computational trust for rumor spreader detection in social media sphere except ours (Rath et al. 2017), where we have conducted a pilot study by utilizing the retweet network of Twitter users. This paper is a natural extension of the original idea by taking into account other types of trust proxies such as reply relationship among the users as well as conducting more thorough experimentations and analyses.

The concept of *Trust* in Twitter's retweet network can be described generally as follows: a user referred to as **A**, who receives a post tweeted from user **B**, may intend to share the post with his/her followers with the action of propagating the information, i.e., *retweet*. **A** might also decide to *reply* the tweeter, which can contain some additional information or comment in the reply beyond a mere retweet. There are two essential factors that can influence the decision of user **A**, who may choose to act on an original post or not: (1) The *trustworthiness* of user **B**, i.e., the willingness of the network to trust **B**; and (2) the *trustingness* of **A**, i.e., the propensity of **A** to trust the other users in the network. According to the prior research of computational trust such as Roy (2015) and Roy et al. (2016), trustingness and trustworthiness are characterized as a pair of complementary measures of user trust in social network and both of them are associated with each network user. A person having higher trustingness contributes to the trustworthiness of its neighbors to a lower degree, while a higher trustworthiness is a result of lots of neighbors linked to the actor having lower trustingness.

Intuitively, users with high trustingness are more likely to spread information online than those with low trustingness since they are more likely to believe what someone else tweets. When the circulated message is false, such users tend to be more likely to become rumor spreaders. On the other hand, users with high trustworthiness are generally less likely to inject or spread false information than those who have low trustworthiness in the sense that the tweets of users who have high trustworthiness are historically retweeted more extensively and they are subjectively more cautious on what they tweet for maintaining their own reputation. As a result, the properties of users in terms of information veracity they are involved in propagating can be inferred somehow based on the nuance of trust relationships among the users.

In this paper, we propose a novel approach for the identification of rumor spreaders based on the concept of *Believability*, which is a measure defined on the basis of trustingness and trustworthiness metrics. Specifically, believability represents the strength of a directed edge between the tweeter **B** and the responder **A**, indicating how strong the potential is for the information from **B** to be spread through **A**. The basic idea is that the believability of the retweeted message is proportional to the trustingness of the responder **A** and the trustworthiness of the tweeter **B**. To this end, we construct the trust network among users, using retweets, and additionally replies, as proxies of trust relationship, for automatically learning the user representation as embeddings in a low-dimension space. More specifically, the representation is inferred from the re-weighted user network with the believability on its edges by employing a state-of-the-art network embedding algorithm called LINE (Tang et al. 2015). Finally, based on the generated user embeddings, we apply supervised machine-learning algorithms such as neural networks or other kind of classifiers to categorize the given user spreading the specific information as a rumor spreader or not.

In a nutshell, the contributions of our paper are threefold:

- To the best of our knowledge, this is the first attempt to identify rumor spreaders on Twitter by exploring the nuance of concepts in computational trust, i.e., trustingness and trustworthiness, for creating a novel measure of believability which quantifies the potential of a message being spread from one user to the others.
- We propose a novel technical framework that strengthens the representation of user properties in consideration of information veracity using network feature learning based on a large-scale believability re-weighted trust network. Experimental results demonstrate the superiority of the proposed method over technically more straightforward approaches.
- We build three Twitter datasets using different trust proxies (i.e., retweet-only, reply-only, retweet+reply) based on a set of real-world rumorous and non-rumorous events gathered from rumor debunking websites, which are made publicly available to research community.[3]

2 Related work

The task of rumor detection can be classified into two categories: rumor information detection and rumor spreader detection. Most of prior research focused on rumor

[3] https://github.com/BhavtoshRath/RNN-Trust/tree/master/data.

information detection. Little work has been done, however, for rumor spreader detection.

Automatic detection of rumorous information from social media is based on traditional classifiers stemming from the pioneering study of information credibility on Twitter (Castillo et al. 2011). In the subsequent studies (Yang et al. 2012; Liu et al. 2015; Ma et al. 2015, 2016, 2017; Wu et al. 2015; Zhao et al. 2015), different sets of hand-crafted features were proposed and incorporated to determine whether a claim about some event is credible or not. However, feature engineering in these methods is painstakingly labor intensive. Ma et al. (2016) proposed a RNN-based method that automatically learns the representations to capture the hidden implications and dependencies of complex signals over time, and achieved better performance due to the effective representation learning capacity of deep neural models. In addition, other neural models such as Convolutional Neural Networks (CNN) and tree-structure Recursive Neural Networks (RvNN) have also been attempted by exploiting either social media content or propagation structures of information in the recent studies (Yu et al. 2017; Ma et al. 2018b). A comprehensive survey focusing on rumor information detection can be found in Zubiaga et al. (2018), and two others focusing on detection on fake news and false information in general can be found in Shu et al. (2017) and Kumar and Shah (2018). In our work, we focus on the rumor spreader detection instead of rumor information detection. To the best of our knowledge, there is no concrete study conducted so far for identifying rumor spreaders via predictive analytics except for a few other works considering spreader characteristics as features for rumor information detection (Kwon et al. 2013; Ma et al. 2015).

Computational trust has been studied extensively in recent years. Many researchers have tried to assign trust scores (Artz and Gil 2007; Kamvar et al. 2003; Mishra and Bhattacharya 2011; O'Hara et al. 2004) to the nodes in a network to accomplish different type of tasks. Trust scores can be defined as scores that an algorithm puts on a node in a trust network based on various structural aspects of the node. Eigentrust (Kamvar et al. 2003) proposes to rate trust scores of peers in a P2P network. These scores can help an ordinary user in the network to identify the trustworthy peers and initiate content download from them. Eigentrust, like Pagerank (Lawrence et al. 1998), calculates a single score for each node in the network. In this algorithm, however, one's reputation does not play a part in the weight of the node's trust vote. Other researchers have proposed measures to rank bias and deserve of a node in a network (Mishra and Bhattacharya 2011), in which they used an iterative matrix algorithm to calculate bias and deserve of nodes which reinforce each other.

Roy (2015) and Roy et al. (2016) proposed a pair of complementary measures that can be used to measure trust scores of actors in a social network using involvement of social networks. Based on the proposed measures, an iterative matrix convergence algorithm based on HITS (Kleinberg 1999) was developed that calculates the trustingness and trustworthiness of each actor in the network. The algorithm runs in $O(k \times |E|)$ time where k denotes the number of iterations and $|E|$ denotes the number of edges in the network. In this paper, we propose a novel measure called *believability* based upon these two complementary measures for assessing the potential of the message being spread from one user to the other, which is used to re-weight the edges of the user trust network. Note that the believability is in essence different from commonly known concept of credibility studied in many papers (Castillo et al. 2011; Gupta et al. 2014; Jin et al. 2014; Metzger and Flanagin 2013), where credibility is primarily used to measure the quality of content being believed in or that of a user being trusted, but believability here is a measure of "spreadability" of information between *a pair of users* instead of an individual user.

Network-based representation learning is an emerging area in machine learning. DeepWalk (Perozzi et al. 2014) learns node embeddings by exploring local neighborhood of the nodes using truncated random walks. Since the strategy of the random walk is uniform following depth-first-search (DFS) style, it gives no control over the explored neighborhoods. Also, it works only for unweighted, undirected graphs. LINE model (Tang et al. 2015) proposes a breadth-first strategy to explore neighborhoods. Specifically, it learns a feature representation in two separate phases: first, it learns half of the dimensions by breadth-first-search (BFS) style simulations over immediate neighbors of nodes, then it learns the other half of dimensions by sampling nodes strictly at a two-hop distance from the source nodes. This model works for all types of graphs. Node2vec (Grover and Leskovec 2016) explores diverse network neighborhoods which designs a sampling strategy that smoothly interpolates between BFS and DFS. The assumption is that BFS and DFS are extreme sampling paradigms suited for structural equivalence (i.e., nodes sharing similar roles) and homophily (i.e., nodes from the sample network community), respectively. Node2vec's sampling strategy accommodates for the fact that these notions of equivalence are not competing or exclusive, and real-world networks commonly exhibit a mixture of both. Considering the weighted, directed nature of our network and the complexity of the learning algorithm, in this paper, we employ LINE algorithm with the two-hop distance for generating user embeddings from the trust network, where the edges are re-weighted by the believability scores.

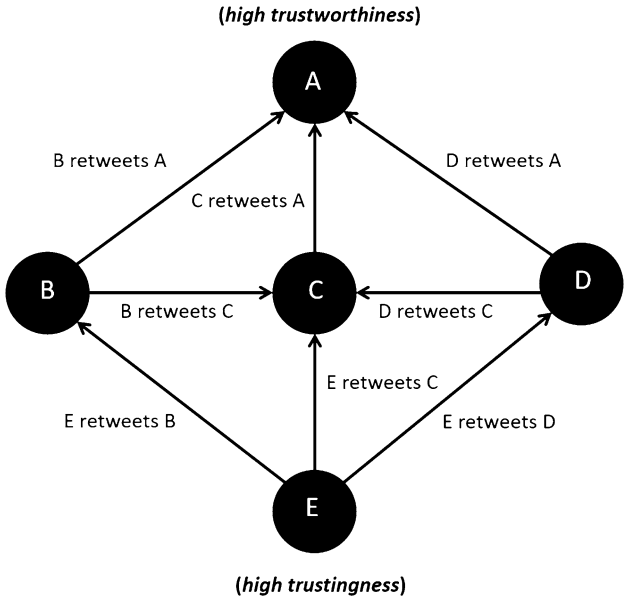

Fig. 1 An illustration of trust in a retweet network, in which the nodes are users and the directed edges indicate the retweet relationship, and each edge can be weighted by the frequency of retweets between two users associated with the edge

3 Trust in social media

Trust is an important part of any social interaction, and in the context of social media, researchers have been using social networks widely to understand how trust manifests among users. However, such an abstract concept of trust that attempts to quantify social interaction is generally very hard to compute. In general, trust in a social network is defined as a set of scores assigned to each actor in the network, representing his/her level of trust.

Specifically, the level of trust can be manifested by assigning a pair of trust scores to each actor which are termed as *trustingness* and *trustworthiness* scores (Roy 2015). Trustingness of an actor is defined as his propensity to trust others in the network. A higher trustingness score implies that the actor has a high propensity to trust others in the network. On the other hand, trustworthiness, being true to its dictionary meaning, defines how trustworthy an actor is which is indicated as the willingness of the network as a whole to trust an individual actor. Like trustingness score, a high trustworthiness score means the trust on the actor from other users in the network is strong.

The primary property leveraged to calculate trust scores is the *negative feedback property* of trust. The concept of negative feedback in trust can be well understood using the example retweet network provided in Fig. 1, which illustrates the trust relationships among the users in a retweet network, where the direction of edges indicates that trust is given to tweeters from retweeters and the number of times of retweeting between two users is used as edge weights. As we can see, there are users, e.g., **E**, which has a high propensity to trust other users as **E** trusts almost all users in the network except node **A**. Thus, it implies that **E** will accord trust to almost anyone in the network which should decrease the weight of its trust vote compared to a node like **C** which accords its trust very selectively. In contrast, it can be seen that node **A** is a highly trusted user since a large number of users in the network retweet its post. Moreover, other nodes such as **B**, **C**, **D** that trust it in turn trusts a very selective amount of other nodes which makes their votes more valuable compared to **E**. Using the negative feedback property, it can be said that a higher trustingness score contributes to the trustworthiness of its neighbors to a lower degree, and a higher trustworthiness score is a result of lots of neighbors having low trustingness scores. Same applies to a reply network.

For quantifying the trust, we require proxies of trust that can map the social interactions to the original concepts of trust. In the context of social network on Twitter, there can be various levels of user interactions acting as the proxies, such as following, retweeting, liking, and replying. For example, a user whose tweets are more likely to be retweeted by others is expected to have a high trustworthiness score, while a user who is more likely to retweet others' tweets is expected to have a high trustingness score. Without the loss of generality, in our work, we adopt retweeting, and additionally, replying interactions as the proxies of trust, and our proposed model is generic and can be straightforwardly extended to accommodate any other kind of proxies.

Algorithm 1 provides the pseudo code of Trust in Social Media (TSM) algorithm for computing trustingness and trustworthness scores (Roy 2015).

Algorithm 1 Trust in Social Media (TSM)

Data: 1. A directed graph G=(V, E) consisting of vertices and edges with or without weights.
2. Maximum number of permitted iterations, k and/or
3. Difference of scores between two iterations, δ.

Result: A set of trust scores: {(trustingness, trustworthiness)} for $\forall\ v \in$V.

Initialize all $v \in$V to (1,1).

for *($i = 1$; $max(max(|ti_i(v) - ti_{i-1}(v)|), max(|tw_i(v) - tw_{i-1}(v)|) < \delta$ or $i \leq k$; i++)* **do**

  for *each node* $v \in V$ **do**

    Update scores of each vertex using scores from last iteration

    $ti_i^{'}(v) = \sum_{\forall out(v)} \frac{w(v,x)}{(1+(tw_{i-1}(x)^{s}))}$

    out(v) = set of all vertices which are destination vertex of all outgoing edges from v;

  end

  for *each node* $v \in V$ **do**

    Update scores of each vertex using scores from last iteration

    $tw_i^{'}(v) = \sum_{\forall in(v)} \frac{w(x,v)}{(1+(ti_{i-1}(x)^{s}))}$

    in(v) = set of all vertices which are source vertex of all incoming edges to v

  end

  $ti_i = Normalize(ti_i^{'})$

  $tw_i = Normalize(tw_i^{'})$

end

3.1 Trustingness and trustworthiness

To calculate trustingness and trustworthiness scores, we use the TSM algorithm (Roy 2015) that takes a directed graph as input together with a specified convergence criteria or a maximum permitted number of iterations. In each iteration for every node in the network, trustingness and trustworthiness are computed using the equations below:

$$ti(v) = \sum_{\forall x \in out(v)} \left(\frac{w(v,x)}{1 + (tw(x))^s} \right), \tag{1}$$

$$tw(u) = \sum_{\forall x \in in(u)} \left(\frac{w(x,u)}{1 + (ti(x))^s} \right), \tag{2}$$

where u and v are user nodes, $ti(v)$ and $tw(u)$ are trustingness and trustworthiness scores of v and u, respectively, $w(v, x)$ is the weight of edge from v to x, $out(v)$ is the set of outgoing edges of v, $in(u)$ is the set of incoming edges of u, and s is the *Involvement* of the network. Involvement of a network is defined as the amount of risk involved in making a wrong link in the network. The higher the risk is in a social network (i.e., the higher the involvement score), the higher the effect of a neighbors trustingness should be on the calculation of the current node trustworthiness, and vice versa. It is thus the potential risk an actor takes when creating a link in the network.

Typically, a user survey can be used to determine the involvement score of different networks, because involvement is a concept inherently perceived by network users. The survey questionnaire can be designed by adopting well-established involvement measurement with a series of seven-point scales. In the survey questionnaire, a detailed description of each social network is provided. Primarily, what each node in the particular social network refers to is listed and also what each link in the network represents. To measure involvement, the respondents are asked questions which are considered proxies for involvement. The questions measured the potential risk of creating a wrong link in the network and the perceived risks associated with the networks. The involvement score for Twitter is set to a constant empirically according to Roy (2015).

Once the trust scores are calculated for each node in the network, TSM normalizes the scores (Roy 2015) by adhering to the normalization constraint so that both the sum of trustworthiness and the sum of trustingness of all nodes in the network equal to 1. However, a salient problem of such normalization method lies in that the scale of the scores is dependent on the size of the network. When the network is very large, the resulting scores will become extraordinarily small. To deal with the issue, we perform min–max normalization based on the logarithm of the scores output by TSM to normalize the trustingness and trustworthiness scores into the range of (0,1].

3.2 Believability

Trustingness and trustworthiness, from different perspectives, are used to measure the level of trust of each individual user. But they do not quantify the strength of *inter-user* trust, i.e., the trust between two specific users who have retweet or reply relationship. The intensity of inter-user trust is very important to indicate the potential or capacity of user network edges for transmitting messages. When a message is propagated, the strength of inter-user trust along the propagation path would largely determine how fast and how far the message could be transmitted over the network. In the original trust model in Fig. 1, edges weighed by the frequency of retweets between two users cannot reflect such kind of "spreadability" of network edges. In this regard, a new method of re-weighting the edges is very much desirable.

We propose the new concept called *Believability*, a quantitative figure that is computed for a directed edge between two nodes used to measure the potential of messages being transmitted through the edge based on the strength of belief between two neighbors on that edge. In the context of retweet, a directed edge from **B** to **A** exists if a tweet of **A** is retweeted by **B**. The believability quantifies the strength that **B** trusts on **A** when **B** decides to retweet **A**. Therefore, **B** is more likely to believe in **A** if:

1. **A** has a high trustworthiness score, i.e., **A** is highly likely to be trusted by other users in the network, or
2. **B** has a high trustingness score, i.e., **B** is highly likely to trust others.

The same applies to the case of reply. So, the believability score is supposed to be proportional to the two values above, which can be jointly determined and computed as follows:

$$Believability(B \rightarrow A) = tw(A) * ti(B). \tag{3}$$

Figure 2 illustrates the relationship between the believability and the trust measures given a retweet or reply edge in the user network. The believability score will be used to re-weight the edges so that the representation of users can be reasonably learned with the differentiation of variable spreadability of different edges. The key reason why this can result in better user representation learning is that the inter-user believability score will lead to the random walk being biased to favorably travel towards nodes via high believability edges (see Sect. 4), thus potentially maximizing the transmission capacity of information over the network.

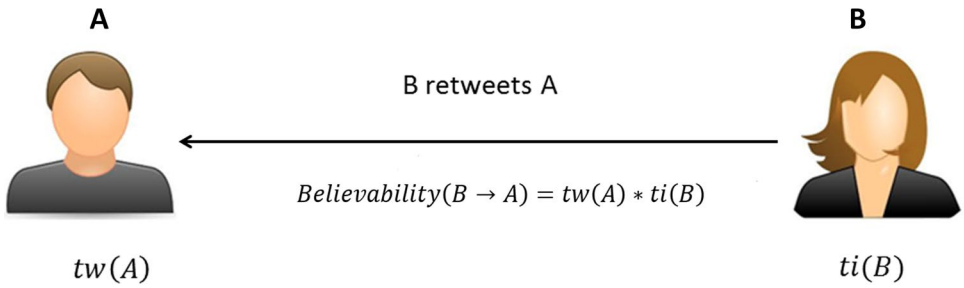

Fig. 2 An illustration of *believability*, which is proportional to the trustworthiness of A and the trustingness of B

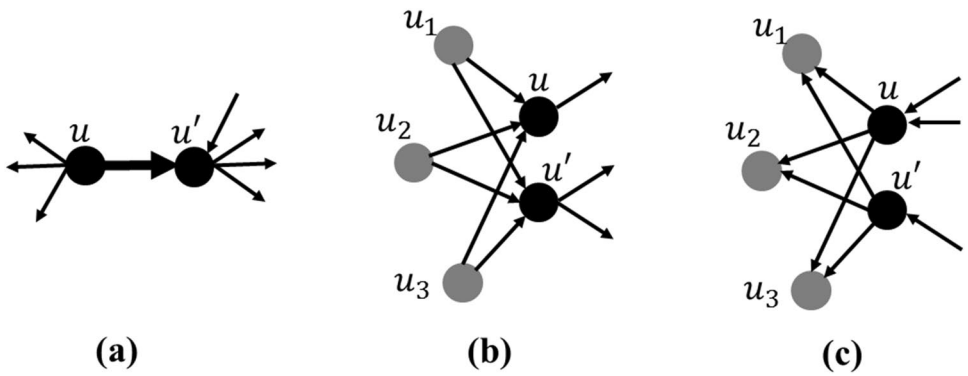

Fig. 3 An illustration of similar users in a network, where u and u' are deemed similar in different contexts. Therefore, u and u' should be projected closely in the representation space

4 User representation learning

In this section, we will discuss how to automatically represent the users based on the re-weighted network using believability scores as the edge weights.

4.1 Rumorous users and context

We define the network as $G = (V, E)$, where $V = \{u_1, u_2, \ldots, u_n\}$ refers to a set of nodes each representing a user, and $E = \{w_{ij}\}$ is a set of directed edges corresponding to the relationship (retweet or reply) among the nodes in V, which are weighted by the believability scores.

Figure 3 illustrates the contexts of network where similar users should be represented closely to each other in the embedding space. Without loss of generality, we illustrate three basic cases of context where two users u and u' reside, which should be considered similar users, and how their similarity is related to rumor propagation:

(a) u and u' act mutually as the context of one another and the u–u' weight is strong, suggesting that u may be a "hard-core fan" of u'. If u' is a frequent rumor spreader, so potentially very likely is u because of the generally low veracity of information provided by u'; and conversely, u' is more likely to be rumorous if u is often rumorous since most of the information u spreading is coming from u'.

(b) u and u' share many common neighbors (such as u_1, u_2, u_3) with in-links, implying that they have a large overlapping group of fans who trust them. If u often pollutes his fans with hearsays while still not losing its audiences, it is very likely that u' is similar to u in terms of information spreading behaviors, because otherwise they could not own many common followers.

(c) Similar but different as (b), u and u' share many neighbors with out-links, indicating that both of them trust in a common group of sources of messages. If u is a frequent receiver of rumors, it is reasonable to infer that u' is inclined to be similar as u because of substantial overlap of their information source or context.

As such, by considering the commonality of context, similar users will be projected closely in the representation space for yielding better classification effectiveness.

4.2 User embeddings

We adopt the second-order proximity between a pair of nodes in a network-based representation learning method (Tang et al. 2015) which is called LINE, to learn user embeddings based on the retweet and reply user network depicted above. The goal is to embed each user $u_i \in V$ into a lower-dimensional space $\mathbb{R}^d$ by learning a function $f_G : V \rightarrow \mathbb{R}^d$, where d is the dimension of the projected vector. Specifically, for each u_i, let $\overline{v}_i$ denote the embedding of u_i as a node and $\overline{v}'_i$ be the representation of u_i when treated as

a specific context of other nodes. For each edge $u_i \rightarrow u_j$, the conditional probability of u_j being generated by u_i as context is defined as follows:

$$p(u_j|u_i) = \frac{\exp(\vec{v}_j'^{\intercal} \cdot \vec{v}_i)}{\sum_{k=1}^{|V|} \exp(\vec{v}_k'^{\intercal} \cdot \vec{v}_i)}. \tag{4}$$

Given this definition, the nodes sharing similar contexts will have similar conditional distributions over the entire set of nodes. To preserve the context proximity, the objective is to make $p(u_j|u_i)$ be close to its empirical distribution $\hat{p}(u_j|u_i)$, where the empirical distribution can be observed from the weighted social context network. Thus, the objective function is defined as:

$$\min \sum_{(i,j)\in E} \lambda_i * d\big(\hat{p}(u_j|u_i), p(u_j|u_i)\big), \tag{5}$$

where $d(\cdot,\cdot)$ is the distance between two probabilities based on KL-Divergence, λ_i is the prestige of u_i which is set to u_i's out-degree d_i following (Tang et al. 2015), and the empirical distribution is computed as $\hat{p}(u_j|u_i) = w_{ij}/d_i$, where w_{ij} is the weight of the edge (i, j).

In the learning, we use LINE[4] for optimizing equation 5, which provides an efficient solution based on negative sampling of edges and asynchronous stochastic gradient descent over the mini-batches of the sampled edges for parameter update.

5 Identifying spreaders of rumors

To identify the rumor spreaders out of a large number users, we use Recurrent Neural Network (RNN) to model the classification process. RNN is used as the classification model for two reasons: first, our data are based on time sequence, i.e. retweets/replies are sequential in nature, where RNN is naturally suitable for the structure of data. Second, the training data are of variable length, i.e., the source tweets can have different number of retweets/replies, for which RNN also fits. It is important to note that there is no fixed time interval between two successive actions. Therefore, we can safely consider that the data are a time sequence instead of time series.

In our experiments (see Sect. 6), we adopt two variants of RNN model: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) model to leverage long-distance dependencies of units in the sequence.

5.1 RNN-based user model

An RNN is a type of feed-forward neural network that can be used to model variable-length sequential information. A basic RNN is formalized as follows: given an input sequence $(x_1, \ldots, x_T)$, for each time step, the model updates the hidden states $(h_1, \ldots, h_T)$ and generates the output vector $(o_1, \ldots, o_T)$, where T depends on the length of the input. From $t = 1$ to T, the algorithm iterates over the following equations:

$$\begin{aligned} h_t &= \tan h(Ux_t + Wh_{t-1} + b), \\ o_t &= Vh_t + c, \end{aligned} \tag{6}$$

where U, W and V are the input-to-hidden, hidden-to-hidden and hidden-to-output weight matrices, respectively, b and c are the bias vectors, and $tanh(.)$ is a hyperbolic tangent nonlinearity function.

Typically, the gradients of RNNs are computed via back-propagation through time (Rumelhart et al. 1986). In practice, because of the vanishing or exploding gradients (Bengio et al. 1994), the basic RNN cannot learn long-distance temporal dependencies with gradient-based optimization. One way to deal with this is to make an extension that includes "memory" units to store information over long time periods, commonly known as Long Short-Term Memory (LSTM) unit (Hochreiter and Schmidhuber 1997; Graves 2013). Gated Recurrent Unit (GRU) (Cho et al. 2014) is another simpler RNN model.

LSTM networks were designed to address the vanishing gradients through a gating mechanism. They are basically an alternative way of computing the hidden state. LSTMs use the following equations to compute the hidden state (Hochreiter and Schmidhuber 1997; Graves 2013):

$$\begin{aligned} i_t &= \sigma(x_t U_i + h_{t-1} W_i), \\ f_t &= \sigma(x_t U_f + h_{t-1} W_f), \\ o_t &= \sigma(x_t U_o + h_{t-1} W_o), \\ g_t &= \tan h(x_t U_g + h_{t-1} W_g), \\ c_t &= c_{t-1} \cdot f_t + g_t \cdot i_t, \\ h_t &= \tan h(c_t) \cdot o_t. \end{aligned}$$

For a basic RNN model, the inputs to the unit were x_t (the current input at time step t) and h_{t-1} (the hidden state in previous time step), and the output was a new hidden state h_t and the output state o_t at current time step. In LSTM, however, the hidden state is computed based on the states of some internal gates using the above equations, which are explained as follows: i_t, f_t, and o_t are the *input*, *forget* and *output* gates computed for time step t. They have the same equations but different parameter matrices. They are called *gates* because the sigmoid function σ converts these into vectors in range between 0 and 1. Multiplying these gates element-wise lets us decide how much of the other vector is let through to the next hidden unit. g_t is a *candidate* hidden state that is computed based on the current input and the previous hidden state. The internal memory c_t is computed

[4] http://github.com/tangjianpku.

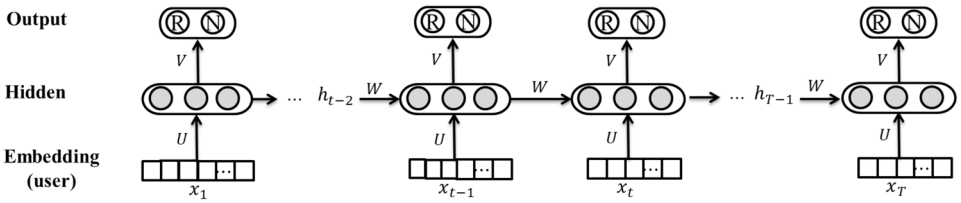

Fig. 4 Our RNN-based model. U, W, V are weight matrices corresponding to the input-to-hidden, hidden-to-hidden and hidden-to-output parameters, respectively. R means the user is a rumor spreading user and N means not a rumor spreading user

as previous memory c_{t-1} multiplied by the *forget* gate, and the newly computed hidden state g, multiplied by the *input* gate. This mechanism allows LSTM to not ignore the old memory completely. We finally compute the output hidden state h_t by multiplying the *hyperbolic tangent* of internal memory with the output gate.

The model of GRU is very similar to that of LSTM layer, however, more simplified. A GRU has gating units that modulate the flow of the content inside the unit, but a GRU is simpler than LSTM with fewer parameters. The following equations are used for a GRU unit in hidden layer (Cho et al. 2014):

$$z_t = \sigma(x_t U_z + h_{t-1} W_z),$$
$$r_t = \sigma(x_t U_r + h_{t-1} W_r),$$
$$\tilde{h}_t = \tan h(x_t U_h + (h_{t-1} \cdot r_t) W_h),$$
$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t,$$

where a reset gate r_t determines how to combine the new input with the previous memory, and an update gate z_t defines how much of the previous memory is cascaded into the current time step, and $\tilde{h}_t$ denotes the candidate activation of the hidden state h_t.

The major difference between LSTM and GRU can be summarized as follows: (1) while an LSTM unit consists of three gates, GRU unit consists of two gates; (2) LSTM unit has an internal memory unit while the GRU unit does not; (3) It is easier to train GRUs than LSTMs as they have fewer parameters. On large datasets, however, LSTMs tend to give better results.

We use the recurrent units of LSTM and GRU to fit the time steps as the basic identification framework. For each source tweet, all of its retweeting or replying users are ordered in terms of the time stamps that indicate when the different users retweet or reply it. In each time step, we input the embedding of the user who retweets or replies the message at that time step. Suppose that the dimensionality of the generated user embedding is K. The structure of our RNN model is illustrated in Fig. 4. Note that an output unit is associated with each of the time steps, which uses *sigmoid* function for the probabilistic output of the two classes indicating the input user is a rumor spreading user or not.

Let g_c, where c denotes the class label, be the ground-truth two-dimensional multinomial distribution of a user. Here, the distribution is of the form [1, 0] for rumor spreading users and [0, 1] for non-rumor spreading users. For each training instance (i.e., each source tweet), our goal is to minimize the squared error between the probability distributions of the prediction and ground truth:

$$\min \quad \sum_c (g_c - p_c)^2 + \sum_i ||\theta_i||^2$$

where g_c and p_c are the gold distribution and predicted distribution, respectively, θ_i represents the model parameters to be estimated, and the L2-regularization penalty is used for trading off the error and the scale of the problem.

5.2 Naive user models

Instead of using a RNN-based user model presented in Sect. 5.1, one may come up with some naive and more straightforward models based upon the property of trust.

5.2.1 Trustingness-only model

Intuitively, users with high trustingness, who easily trust others, are more likely to spread rumors. Our trustingness-only model simply learns a threshold based on the correlation between the trustingness score and ground truth of users in the training data. With the threshold, the model can easily predict user class given the trustingness of a new user. The model is described as follows:

$$\text{prediction}(u) = \begin{cases} \text{true} & \text{if } \textit{trustingness}(u) \geq \mathcal{T}_{ti}; \\ \text{false} & \text{otherwise} \end{cases} \quad (7)$$

where $\mathcal{T}_{ti}$ is the threshold of trustingness score to be learned from training.

5.2.2 Trustworthiness-only model

In contrast, the users with high trustworthiness who are more trustworthy are less likely to spread rumors. The trustworthiness-only model similarly learns a threshold from the training data capturing the relationship between

the trustworthiness score and ground truth label of users. Similar to Eq. 7, the trustworthiness-only model is given as below:

$$\text{prediction}(u) = \begin{cases} \text{false} & \text{if } \textit{trustworthiness}(u) \geq \mathcal{T}_{tw}; \\ \text{true} & \text{otherwise.} \end{cases} \tag{8}$$

where $\mathcal{T}_{tw}$ is the threshold of trustworthiness score to be learned from training data.

5.2.3 Interpolation model

The interpolation model linearly combines the trustingness and trustworthiness scores in such a way that they are interpolated with the appropriate weights to give an optimal prediction on its trust score. The trust score of a given user can be predicted as:

$$T(u) = \alpha * \textit{trustingness}(u) + (1 - \alpha) * \textit{trustworthiness}(u)$$

where α is the weight that can be fixed during training stage. With the similar thresholding strategy above, we can obtain the threshold $\mathcal{T}_{tr}$ of the interpolated trust score, and the class of user can be predicted as follows:

$$\text{prediction}(u) = \begin{cases} \text{true} & \text{if } T(u) \geq \mathcal{T}_{tr}; \\ \text{false} & \text{otherwise.} \end{cases} \tag{9}$$

5.2.4 Logistic regression classifier with user embeddings

For machine-learning-based approach, in addition to use RNN-based model, we can also utilize other type of classifiers such as logistic regression or support vector machines for classifying the users. The user embeddings generated from the LINE algorithm can be straightforwardly treated as features of the classifiers.

Here we use the L2-regularized logistic regression (L2_LR) classifier (Fan et al. 2008). The L2_LR solves the following unconstrained optimization problem:

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \log\left(1 + \exp\left(-y_i w^T x_i\right)\right), \tag{10}$$

where x_i and y_i are the input vector of an instance and its prediction, and w is the vector of model parameters to learn. The reasons why we have chosen L2_LR are that (a) it has consistently delivered state-of-the-art performance in several applications (Conroy and Sajda 2012; Zou et al. 2015), and is thus a strong contender, and (b) logistic regression, like RNN, natively outputs posterior probabilities (Murphy 2012), which is comparable.

6 Experiments and results

In this section, we will describe the collection of datasets, comparative experiments and the results achieved.

6.1 Data collection

We constructed our datasets based on two reference datasets, namely Twitter15 (Liu et al. 2015) and Twitter16 (Ma et al. 2016), which were previously used for binary classification of rumor and non-rumor with respect to a given event that contains its relevant tweets. The two Twitter datasets were originally constructed by first gathering a set of rumorous and non-rumorous events from rumor debunking websites such as www.snopes.com. For every event on the debunking websites, there is a main claim about the specific event which is associated with it. Then they gathered the relevant tweets of each event via keyword search on Twitter's site, for which a set of keywords were manually composed based upon the claim of each event (Liu et al. 2015; Ma et al. 2016).

In our work, given the main claim of each event in the two datasets, we extracted from them the popular source tweets that are highly retweeted or replied. We then constructed the propagation threads (i.e., retweets and replies) for these source tweets. Because Twitter API cannot retrieve the retweets and replies, we gathered the retweet users for a given tweet from Twrench.[5] We also crawled the replies to the source tweets through Twitters web interface. Note that each main claim is associated with a gold-standard veracity label, i.e., rumor or non-rumor, derived directly from the rumorous and non-rumorous events on the rumor debunking websites. The label of the claim associated with event will be used to generate the ground truth of users for rumor spreader detection (see Sect. 6.1.2).

6.1.1 User trust network

Based on the users appearing in these events, we constructed three networks (i.e., retweet-only, reply-only and retweet+reply) using the following steps:

1. We merged the two reference datasets into one large corpus;
2. We obtained the follow relationships among the users that have appeared across all the events for getting an initial user network.[6] In particular, we treat the follow relationship on Twitter as a special yet rudimentary form

[5] https://twren.ch.

[6] We used Twitter API for getting maximum 5k friends of each user, and obtained more friends by requests via Twitter's Web interface.

Table 1 Statistics of user trust networks

	Retweet	Reply	Combined
Total # of nodes	1,292,708	1,122,797	1,321,872
Total # of edges	19,533,330	18,535,341	19,645,380
Avg in-degree	15.1	16.5	14.9
Max in-degree	95,303	95,302	95,303
Min in-degree	0	0	0
Avg out-degree	15.1	16.5	14.9
Max out-degree	52,519	6,740	58,274
Min out-degree	0	0	0

of retweet with frequency of 1 (i.e., a follow relationship is counted as one-time retweet) to alleviate the sparsity of the generated retweet network.

3. From each of the events, we extracted popular source tweets with more than 50 retweets and replies altogether[7] that are highly responded;
4. We collected all the retweet users for each source status.[8] These retweet relationships are added as edges into the initial users network above to form the retweet-only network.
5. We then collected all the reply users for each source status.[9] These reply relationships are added as edges into the initial user network to form the reply-only network.
6. We finally generate the retweet+reply network by joining the two networks obtained above.

The statistics for the three networks are shown in Tables 1. These three user networks will be used to compute the user trust scores for then generating the user embeddings (see Sects. 3 and 4).

6.1.2 User classification dataset

We also built user classification dataset for our RNN models (see Sect. 5.1) based on the source tweets, where each source tweet is associated with a sequence of retweeters/repliers ordered by the time stamp when the retweet/reply occurs.

The ground-truth label for each user is determined by the nature of the source tweet by following these rules:

1. If the main claim of event is reported as rumorous and the source tweet supports it, the ground truth of the users participating in spreading it is labeled as rumor spreaders; if the source tweet denies the claim, the users participating in spreading it are labeled as non-rumor spreaders.
2. If the main claim is reported as a non-rumor, the ground truth labels of the users responding to the source tweet are assigned in an opposite way as the rumor case above depending on the specific stance of the source tweet that the users are participating in spreading.
3. When a user appears both in rumor and non-rumor cases, we check the frequency that the user has appeared in the two cases: if it appears more often in rumor than in non-rumor cases, it is labeled as a rumor spreader; otherwise labeled as a non-rumor spreader.

Table 2 Statistics of user classification datasets

	Retweet	Reply	Combined
# of unique users	902,806	98,373	969,857
# of users spreading rumors	417,569	49,671	415,846
# of users not spreading rumors	485,237	48,702	554,011
Total # of source tweets	3098	3068	3098
# of rumor source tweets	1716	1690	1716
# of non-rumor source tweets	1382	1378	1382
Avg # length of source tweet sequences	413.98	52.48	464.9
Max # length of source tweet sequences	2915	814	3145
Min # length of source tweet sequences	49	2	56

According to these rules, the procedure of ground truth assignment was carried out semi-automatically, in which only the stance of source tweets need to be determined manually.

The statistics on the user classification datasets are shown in Tables 2. As we can observe, there are few major distinctions between the retweet and reply dataset generated, which may contribute to the differences in the experimental results reported in later section. These differences are described as follows:

- The reply dataset is much smaller than the retweet dataset (both in terms of network size and average sequence length). This is due to the fact that the most popular interactions among users on Twitter are retweet rather than reply, which is reflected properly in our data.
- While each sequence in the retweet dataset has different users, this need not be the case with the reply dataset, as few users in the dataset can participate in the conversation chain via reply multiple times.
- Reply and retweet are both indicators of active engagement with information exchange. We believe that retweet

[7] Though unpopular tweets could be fake, we ignore them as they do not draw much attention and are hardly impactful.

[8] Since Twitter API cannot retrieve over 100 retweets, we gathered the retweet users for a given tweet from Twrench (https://twren.ch).

[9] We generated the replies of the source tweets using PHEME toolkit (https://github.com/azubiaga/pheme-twitter-conversation-collection; Zubiaga et al. 2015).

is, however, a stronger indication of trust than reply. This is because: (1) through a retweet, a user basically causes information broadcasting to all its followers, and in contrast, a reply is not broadcasted, and remains confined to the conversation chain of the source tweet. (2) A retweet contains nothing but the original post, implying a supportive stance on source tweet. A reply, however, can contain additional text expressing a different attitude toward the source message.

6.2 Settings and protocols

We ran TSM to get the trust scores based on our networks, which is then re-weighted by the believability scores. We adopted the generic setting of TSM involvement parameter $s = 0.391$ by referring to Roy (2015). Then, we learned the user embeddings in the networks by running LINE algorithm, where we empirically set the size of embeddings as 200 and kept other parameters as the default settings.

For user classification, we fed the sequence of users of each source tweet into RNN's input layer one at a time and trained the RNN model by employing the derivative of the loss with respect to all the parameters via back propagation (Collobert et al. 2011). We used gradient descent for parameter update. The size of the hidden units is set as 100 and the learning rate as 0.005, and the number of epoch as 100 for ensuring the convergence of RNN. In prediction, the probabilities of the same users if they appear across different source tweets are averaged for predicting the final class labels.

We made systematic comparisons among the following eight models:

- *Trustingness* The trustingness-only user model (Sect. 5.2.1);
- *Trustworthiness* The trustworthiness-only user model (Sect. 5.2.2);
- *Interpolation* The model that interpolates the trustingness-only and trustworthiness-only models (Sect. 5.2.3);
- *L2_LR* The L2-regularized logistic regression classifier using user embeddings as features (Sect. 5.2.4);
- *GRU_noweight* The GRU-based RNN user model using user embeddings as features which were obtained from the unweighted user network whose edge weights are all set equal to 1;
- *GRU_notrust* The GRU-based RNN user model using user embeddings as features which were obtained from the initial user network (with only follow relationship) without considering other trust relationship;
- *GRU_trust* The GRU-based RNN user model using user embeddings as features which were obtained from the user network whose edges are re-weighted with believability scores.
- *LSTM_trust* The LSTM-based RNN user model using user embeddings as features which were obtained from the user network whose edges are re-weighted with believability scores.

For evaluation purpose, we ran experiments based on tenfold cross-validation and used five common evaluation metrics: accuracy, area under the receiver operating characteristic curve (AUROC), precision, recall and F1 measure. The accuracy is defined over the two classes as: $\text{Accuracy} = \frac{\#\text{ of correctly predicted users}}{\text{Total }\#\text{ of users}}$. AUROC is the area under the graphical plot which combines true positive rate and false positive rate into one metric and provides a measure of performance (between 0 and 1) across all possible classification thresholds. In all our models we have used the classification threshold of 0.5. The rest of the three metrics are defined for each individual class: for the positive class, i.e., rumor spreading users, the precision is defined as $\text{Precision}\,(+) = \frac{\text{FP}}{\text{TP+FP}}$, the recall is defined as $\text{Recall}\,(+) = \frac{\text{TP}}{\text{TP+FN}}$, and F_1 is defined as $F_1 = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision+Recall}}$, where TP, FP and FN are true positive rate, false positive rate and false negative rate, respectively. The corresponding metrics for the negative class, i.e., non-rumor spreading users, are defined similarly.

7 Results and analysis

We now provide a detailed description and analysis of the experimental results for the three datasets in the following three sections.

7.1 Retweet-only dataset

Table 3 shows experimental results run on the retweet-only dataset. It is observed that **Trustingness** model has an accuracy of 0.538. **Trustworthiness** model shows a marginal increase in accuracy 0.56%. AUROC for **Trustworthiness** model also shows a minor increase of 5.2% over **Trustingness** model. While the precision of rumor class noticeably drops by 25.7%, the recall of the same class increases sharply by 800%. Other metric values remain almost identical. This is because the trustworthiness scores are much better scaled than the trustingness scores. The network is very sparse with a large number of star-shaped sub-graphs. This means that a large portion of the nodes have no outgoing edges, which results in lots of identical trustingness scores. However, the two models are basically comparable, which is attributed to the fact that the two scores are complementary measures derived from a global user interaction network which are essentially the reciprocal sides of trust. So, overall they contribute equally to the spreader detection.

Table 3 Result comparison of different identification models on the retweet-only dataset

Method	Model	Class	Accuracy	AUROC	Precision	Recall	F_1
Naive	TN	+	0.538	0.498	0.847	0.002	0.003
		−			0.538	0.999	0.699
	TW	+	0.541	0.524	0.629	0.018	0.036
		−			0.540	0.990	0.699
	TNnTW	+	0.539	0.513	0.609	0.016	0.032
		−			0.538	0.990	0.697
	L2_LR	+	0.537	0.500	0.891	0.000	0.001
		−			0.537	0.999	0.698
RNN	GRU_NW	+	0.618	0.655	0.555	0.478	0.513
		−			0.654	0.721	0.686
	GRU_NT	+	0.592	0.661	0.536	0.249	0.340
		−			0.606	0.842	0.705
	GRU_TR	+	0.664	0.716	0.590	0.664	0.625
		−			0.731	0.663	0.695
	LSTM_TR	+	0.698	0.731	0.626	0.706	0.664
		−			0.764	0.692	0.726

'+' denotes rumor spreading user, and '−' denotes non-rumor spreading user (TN: trustingness; TW: trustworthness; TNnTW: interpolation; L2_LR: L-2 regularized logistic regression; GRU_NW: GRU_noweight; GRU_NT: GRU_notrust; GRU_TR: GRU_trust; LSTM_TR: LSTM_trust)

The **Interpolation** model which combines the above two models shows similar results. Since the interpolation method just adopts a linear combination of the trustingness and trustworthiness scores, it is cannot capture any feature-level correlation, and as expected its performance lies in-between, with an accuracy of 0.539 and an AUROC of 0.513. In comparison with **Trustworthiness** model, while the metrics for the non-rumor class show almost identical results, rumor class shows slight decrease in all metrics.

L2_LR, which uses as features the embeddings based on believability-weighted edge scores, also shows similar results. The AUROC score suggests that the model acts as a random predictor. This is a bit surprising since it appears that L2_LR which is a learning-based model fails to learn useful patterns from the generated embeddings. There could be reasons from two aspects: (1) a point to notice in baseline results is that both recall and F_1 scores are very low for rumor class. It shows a bias towards predicting most classes as non-rumor spreaders, resulting in very low *True Positives* and *False Positives*, which could be attributed to the fact that the edge-density ratio of the network is very low due to the sparsity of the network. This caused the trust scores to be less scaled. (2) Using such embeddings resulting from a sparse network, L2_LR, although being learning-based, does not consider higher level abstractions or correlations among these features, and also cannot naturally take into account the dependencies among the users in the sequence, rendering model's generalizability as low as the simple **Interpolation** model.

The **GRU_noweight** model employs the RNN algorithm for user classification but does not take into consideration the different strength of the retweet edges. The accuracy for the model is 0.618, which gives around 15.1% improvement over **L2_LR** model. AUROC for the model is 0.655, a significant improvement of 31% over the L2_LR model. More importantly, recall and F_1 scores on both classes show a significant improvement over baseline results. Thus we can conclude that the user representation learning based on even unweighted retweet relationships together with RNN classification improves the ability to identify rumor spreaders. This can be attributed to the strong learning capacity of RNN-based models by the use of hidden units capturing complex feature correlations and the consideration of long-distance temporal dependencies among the users.

The **GRU_notrust** model takes into account the basic following relationship while the **GRU_noweight** considers the unweighted retweet network. The accuracy is improved over the **L2_LR** by 10.2% to achieve 0.592. This can be attributed to the fact that the follow relationship network is sparser than the retweet network, which is a sub-graph of the retweet relationship network. However, **GRU_notrust** shows a minor decrease of 4.2% in accuracy and a slight increase of 0.91% in the AUROC score over **GRU_noweight** model.

The **GRU_trust** and **LSTM_trust** models consider the believability scores for the retweet edges based on the complementary trust measures derived from the overall topology of the network. The accuracy for **GRU_trust** is

Table 4 Results comparison of different identification models on the reply-only dataset)

Method	Model	Class	Accuracy	AUROC	Precision	Recall	F_1
Naive	TN	+	0.506	0.501	0.506	0.999	0.672
		−			0.499	0.000	0.001
	TW	+	0.534	0.524	0.519	0.977	0.678
		−			0.790	0.087	0.156
	TNnTW	+	0.532	0.524	0.518	0.979	0.678
		−			0.797	0.08	0.145
	L2_LR	+	0.502	0.499	0.502	0.998	0.668
		−			0.526	0.002	0.004
RNN	GRU_NW	+	0.544	0.591	0.696	0.124	0.210
		−			0.529	0.948	0.679
	GRU_NT	+	0.546	0.594	0.646	0.164	0.262
		−			0.532	0.913	0.672
	GRU_TR	+	0.612	0.689	0.561	0.146	0.232
		−			0.618	0.923	0.741
	LSTM_TR	+	0.649	0.709	0.581	0.438	0.499
		−			70.678	0.789	0.729

'+' denotes rumor spreading user, and '−' denotes non-rumor spreading user (TN: trustingness; TW: trustworthness; TNnTW: interpolation; L2_LR: L-2 regularized logistic regression; GRU_NW: GRU_noweight; GRU_NT: GRU_notrust; GRU_TR: GRU_trust; LSTM_TR: LSTM_trust

further improved over **GRU_notrust** by 12.1% and reaches 0.664. Also, its accuracy improves over **GRU_noweight** and **L2_LR** models by 7.4% and 23.6%, respectively. In addition, the AUROC of **GRU_trust** increases by 8.3% over GRU_notrust to reach 0.716, and the AUROC for **LSTM_trust** further increases to 0.731. In terms of the *precision*, *recall* and F_1 measure, the performances on both classes also demonstrate consistent improvement. **LSTM_trust** performs better than **GRU_trust** with an improvement of 5.1% in terms of accuracy. Compared to the GRU model, it also shows improvement in all other metric values for both classes. This is because the LSTM model uses an additional memory unit which captures the temporal dependencies better than the GRU model. The higher performance of these two RNN-based models indicate that not only using as input the user embeddings derived from the believability-re-weighted retweet network is advantageous, but also when the embeddings are used in a RNN framework that takes into consideration temporal dependencies, we can improve the final classification effectiveness on users.

7.2 Reply-only dataset

The reply dataset shows some interesting results compared to the retweet dataset. In Table 4, the **Trustworthiness** model shows an increase of 5.5% in accuracy and 4.6% in AUROC score than the **Trustingness** model, which suggests that trustworthiness score is a better metric than trustingness to classify users based on reply behavior. This can be explained as the fact that the reply network, similar to retweet network being very sparse, is also relatively smaller. In such a sparse, relatively small network, many star-shaped sub-graphs exist, with many nodes having no outgoing edges. As a result, those nodes have the same trustingness scores rendering trustworthiness scores are better scaled than trustingness scores.

The **Interpolation** model, not surprisingly, shows a similar effect on accuracy as the retweet-only case, i.e., its performance lies between the two individual models. In terms of other metrics, it performs similarly as the **Trustworthiness** model, which indicates that the simple interpolation without deeper integration cannot boost the performance further.

Noticeably, the **L2_LR** model performs even worse than all other baselines with some larger margin than the retweet-only case, similar to a random guessing. This indicates that, being contrary to the general perception, learning is not helpful in this particular case. We attribute the poor performance of **L2_LR** to two reasons: (1) the reply-only network is sparser and relatively smaller, which results in relatively weaker user embeddings; (2) more importantly, unlike RNN model, L2_LR learns the model based on individual user embeddings and cannot easily consider other users in the context of information propagation, rendering a weak predicative model.

An interesting observation in the baseline model statistics is the poor recall and F_1 score of the non-rumor class. It shows a bias towards predicting most classes as rumor

Table 5 Result comparison of different identification models on the combined (retweet and reply) dataset

Method	Model	Class	Accuracy	AUROC	Precison	Recall	F_1
Naive	TN	+	0.551	0.612	0.533	0.289	0.376
		−			0.558	0.779	0.650
	TW	+	0.536	0.551	0.584	0.012	0.023
		−			0.535	0.992	0.695
	TNnTW	+	0.553	0.574	0.537	0.295	0.381
		−			0.558	0.778	0.650
	L2_LR	+	0.629	0.647	0.602	0.602	0.602
		−			0.652	0.652	0.652
RNN	GRU_NW	+	0.679	0.704	0.667	0.480	0.559
		−			0.684	0.824	0.748
	GRU_NT	+	0.705	0.715	0.702	0.526	0.601
		−			0.707	0.836	0.767
	GRU_TR	+	0.724	0.740	0.745	0.529	0.619
		−			0.716	0.867	0.784
	LSTM_TR	+	0.738	0.795	0.758	0.559	0.643
		−			0.729	0.869	0.793

'+' denotes rumor spreading user, and '−' denotes non-rumor spreading user (TN: trustingness; TW: trustworthness; TNnTW: interpolation; L2_LR: L-2 regularized logistic regression; GRU_NW: GRU_noweight; GRU_NT: GRU_notrust; GRU_TR: GRU_trust; LSTM_TR: LSTM_trust)

spreaders, resulting in very low *True Negatives* and *False Negatives*, which is the opposite case of the retweet-only dataset (see Sect. 7.1). This could be attributed to the fact that compared to retweeting, where a user chooses to broadcast a tweet to all its followers, replying may not be a very strong metric of trustingness because it could imply various types of stances such as supporting, against or neutral.

Similar to the results on the retweet-only dataset, using RNN-based models show significant improvement in accuracies and AUROC score. The accuracy of **GRU_noweight** model is 0.544, an increase of 8.3% over previous baseline. The AUROC score also shows an increase of 18.4%. **GRU_notrust** shows similar trend of improving results as the **GRU_noweight** model. Introducing Trust property in **GRU_trust** model improved accuracy by over 12% and AUROC by 15.9%. Interestingly, all the GRU models have low Recall and F_1 score for the rumor class. The **LSTM_trust** model shows the best results with the highest accuracy of 0.649, the highest AUROC score of 0.709, and more balanced metric values for both the classes.

7.3 Combined (retweet + reply) dataset

The retweet and reply datasets for user classification are merged to form this dataset, in which the retweeting and replying user sequences of the same source tweets are concatenated according to the sequence of time stamps of the user posting the responses. Experiments performed on this dataset also show interesting results.

Table 5 shows that for the first time the **Trustingness** model performs better than the **Trustworthiness** model, with an increase of 2.7% in accuracy and 11% in AUROC score. By examining the precision and recall specifically, we find that the improvement of the **Trustingness** model primarily comes from the significant rise of recall of rumor classes with only some minor drops on the precision of rumor class and the recall of non-rumor class. As shown in Table 2, the combined network is less sparse with more inter-connected nodes. It suggests that as most of the retweeter/replier nodes also have outgoing edges, the trustingness scores are better scaled in this case.

The **Interpolation** model also shows an improvement in all metrics. This can be attributed to the fact that both of the trust scores are better scaled due to the improved connectivity of the user network.

The **L2_LR** model shows an improvement of 13.7% in accuracy and 12.7% in AUROC score over the **Interpolation** model. Further observation on the precision and recall unveils that significant improvement lies in the rumor class where the F_1 is boosted by 58%. This indicates that the embeddings are much better due to the improved user network density giving rise to the better scaled trust scores.

Similar to Table 3, RNN-based models show significant improvement in performance. The accuracy for **GRU_noweight** model is 67.9%, giving an improvement of 7.9% over previous best baseline model. The model also shows a high AUROC score of 0.704. We can also observe that **GRU_notrust** outperforms **GRU_noweight** with 3.83%

improvement in accuracy and 1.6% improvement in AUROC score, suggesting that the initial user network with only follow relationship is helpful since user's following behavior, although not strong, is a kind of, maybe the simplest form of, implication of user trust. In addition, the other two more advanced neural network models, i.e., **GRU_trust** and **LSTM_trust**, unsurprisingly make further improvement on the overall performance. The **LSTM_trust** model gives the best results in our experimental setup with an accuracy of 0.738 and AUROC score of 0.795. This can be attributed to the fact that the combined dataset is larger and the user network is denser.

7.4 Effect of change in hyper-parameters

Furthermore, we studied the influence of the hyper-parameters of TSM algorithm, i.e., the involvement score, and the LINE algorithm, i.e., vector length of embeddings. We found, however, that there was no significant variations in any of the evaluation metric values. This indicates that our model is not sensitive to the setting of these hyper-parameters. Therefore, we do not intend to present these marginally different results in the paper.

In addition, we also studied the effect of sparse user networks by considering alternative proxies such as '1 mention', '1 reply', and '1 retweet' without sparsity relief by excluding follow relationship, which leads to an edge connecting two users if there is one '@' mention, one reply or one retweet between them. However, the results are overall discouraging: the naive approaches were giving close to 50% performance in accuracy, and the RNN-based models gave only a slight improvement of around 55%. Therefore, we did not demonstrate them here in detail.

8 Conclusions and future work

In this paper, we conducted a pilot study for the identification of rumor spreading users on Twitter based on computational trust measures. We proposed a machine-learning framework using the novel concept of believability which is defined based upon the trust measures of individual users in a large-scale retweet network. The key hypotheses are that: (1) the believability between two users is proportional to the trustingness of the responder and the trustworthiness of the source, where trustingness and trustworthiness are two complementary trust measures inferred from users' behaviors; (2) in return, using the believability for edge reweighting on the networks can help enhance the learning of feature representation of users in the network, whereby the users' structural properties can be better preserved in terms of neighborhood similarity, signaling the distinctive roles different types of users play in spreading messages. We proposed LSTM- and GRU-based RNN models for user classification using user embeddings as input features that are generated from the believability re-weighted retweet network. Experimental results on a large real-world user classification dataset collected from Twitter demonstrate that the proposed RNN-based method outperformed four more straightforward methods with large margin.

The research work could be used to build Social Media Reputation framework (similar to how feedback scores for eBay users is calculated). We can associate a trust score to the users in social media that would let service providers to authenticate the veracity of information. Low trust users when detected can be monitored to prevent any future occurrence of rumor propagation. Thus this research can be used to make social media a more veracious source of information.

Overall, the performance on detecting rumor spreaders is not very high, indicating the task is difficult. In the future, we plan to extend our model by incorporating additional proxies of trust more than just retweets and replies. The stance of a user who replies the message is an important indicator as to whether the user is a rumor spreader or not. We will try to directly model user stances with deeper understanding of the concerned user's profile and interactions in the past for inferring the hidden intent and motivation of users retweeting a message.

We would also like to make a distinction between regular, non-regular users and bots to study their rumor spreading characteristics. We shall enhance our data collection to alleviate the sparsity of user trust networks which seems an important issue. In addition, we propose to study rumor information detection based on user trust networks and compare it with state-of-the-art detection systems. Meanwhile, we would be interested to investigate how to perform multiple detection tasks in rumorous environment such as detecting rumors and their spreaders at the same time.

References

Allport G, Postman L (1965) The psychology of rumor. Russell & Russell, Westerham

Arif A, Shanahan K, Chou F, Dosouto Y, Starbird K, Spiro E (2016) How information snowballs: exploring the role of exposure in online rumor propagation. In: Proceedings of CSCW. ACM, pp 466–477

Artz D, Gil Y (2007) A survey of trust in computer science and the semantic web. Web Semant Sci Serv Agents World Wide Web 5(2):58–71

Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 5(2):157–166

Castillo C, Mendoza M, Poblete B (2011) Information credibility on twitter. In: Proceedings of WWW. ACM, pp 675–684

Cho K, van Merrienboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: encoder–decoder approaches. arXiv:1409.1259

Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537

Conroy BR, Sajda P (2012) Fast, exact model selection and permutation testing for L2-regularized logistic regression. In: Proceedings of the 15th International Conference on Artificial Intelligence and Statistics, pp 246–254

Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J (2008) LIBLINEAR: a library for large linear classification. J Mach Learn Res 9:1871–1874

Graves A (2013) Generating sequences with recurrent neural networks. arXiv:1308.0850

Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of SIGKDD. ACM, pp 855–864

Gupta A, Kumaraguru P, Castillo C, Meier P (2014) TweetCred: real-time credibility assessment of content on Twitter. In: Proceedings of SocInfo, lecture notes in computer science, vol 8851. Springer, Cham, pp 228-243

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

Jin Z, Cao J, Jiang Y, Zhang Y (2014) News credibility evaluation on microblog with a hierarchical propagation model. In: Proceedings of ICDM. IEEE, pp 230–239

Kamvar SD, Schlosser MT, Garcia-Molina H (2003) The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of WWW. ACM, pp 640–651

Kleinberg J (1999) Authoritative sources in a hyperlinked environment. J ACM 46(5):604–632

Kumar S, Shah N (2018) False information on web and social media: a survey. CoRR. arxiv:abs/1804.08559

Kwon S, Cha M, Jung K, Chen W, Wang Y (2013) Prominent features of rumor propagation in online social media. In: Proceedings of ICDM

Lawrence P, Sergey B, Motwani R, Winograd T (1998) The pagerank citation ranking: bringing order to the web. Technical report. Stanford University

Liu X, Nourbakhsh A, Li Q, Fang R, Shah S (2015) Real-time rumor debunking on twitter. In: Proceedings of CIKM. ACM, pp 1867–1870

Ma J, Gao W, Wei Z, Lu Y, Wong K-F (2015) Detect rumors using time series of social context information on microblogging websites. In: Proceedings of CIKM. ACM, pp 1751–1754

Ma J, Gao W, Mitra P, Kwon S, Jansen BJ, Wong K-F, Meeyoung C (2016) Detecting rumors from microblogs with recurrent neural networks. In: Proceedings of IJCAI. AAAI, pp 3818–3824

Ma J, Gao W, Wong K-F (2017) Detect rumors in microblog posts using propagation structure via kernel learning. In: Proceedings of ACL. Association for Computational Linguistics, pp 708–717

Ma J, Gao W, Wong K-F (2018a) Detect rumor and stance jointly by neural multi-task learning. In: Companion proceedings of the web conference. ACM, pp 585–593

Ma J, Gao W, Wong K-F (2018b) Rumor detection on twitter with tree-structured recursive neural networks. In: Proceedings of ACL. Association for Computational Linguistics, pp 1980-1989

Metzger MJ, Flanagin AJ (2013) Credibility and trust of information in online environments: the use of cognitive heuristics. J Pragmat 59:210–220

Mishra A, Bhattacharya A (2011) Finding the bias and prestige of nodes in networks based on trust scores. In: Proceedings of WWW. ACM, pp 567–576

Murphy KP (2012) Machine learning: a probabilistic perspective. The MIT Press, Cambridge

O'Hara K, Alani H, Kalfoglou Y, Shadbolt N (2004) Trust strategies for the semantic web. In: Proceedings of the 2004 International conference on trust, security, and reputation on the semantic web, vol 127. Aachen, pp 42–51

Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of SIGKDD. ACM, pp 701–710

Rath B, Gao W, Ma J, Srivastava J (2017) From retweet to believability: utilizing trust to identify rumor spreaders on Twitter. In: Proceedings of ASONAM. ACM/IEEE, pp 179–186

Roy A (2015) Computational trust at various granularities in social networks. Ph.D. thesis, University of Minnesota

Roy A, Sarkar C, Srivastava J, Huh J (2016) Trustingness and trustworthiness: a pair of complementary trust measures in a social network. In: Proceedings of ASONAM. CM/IEEE, pp 549–554

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323:533–536

Shu K, Sliva A, Wang S, Tang J, Liu H (2017) Fake news detection on social media: a data mining perspective. SIGKDD Explor 19(1):22–36

Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) LINE: large-scale information network embedding. In: Proceedings of WWW. ACM, New York, pp 1067–1077

Wang S, Terano T (2015) Detecting rumor patterns in streaming social media. In: Proceedings of the IEEE international conference on big data. IEEE, pp 2709-2715

Wu K, Yang S, Zhu KQ (2015) False rumors detection on sina weibo by propagation structures. In: Proceedings of ICDE. IEEE, pp 651–662

Yang F, Liu Y, Yu X, Yang M (2012) Automatic detection of rumor on sina weibo. In: Proceedings of ACM SIGKDD workshop on mining data semantics. ACM, New York

Yu F, Liu Q, Wu S, Wang L, Tan T (2017) A convolutional approach for misinformation identification. In: Proceedings of IJCAI. AAAI, pp 3901–3907

Zhao Z, Resnick P, Mei Q (2015) Enquiring minds: early detection of rumors in social media from enquiry posts. In: Proceedings of WWW. ACM, pp 1395–1405

Zou F, Wang Y, Yang Y, Zhou K, Chen Y, Song J (2015) Supervised feature learning via L2-norm regularized logistic regression for 3D object recognition. Neurocomputing 151:603611

Zubiaga A, Aker A, Bontcheva K, Liakata M, Procter R (2018) Detection and resolution of rumours in social media: a survey. ACM Comput Surv 51(2):32

Zubiaga A, Hoi GWS, Liakata M, Procter R, Tolmie P (2015) Analysing how people orient to and spread rumours in social media. PLoS One 11(3):e0150989